TD Option informatique $n°4$ – Mots de Lyndon et de de Bruijn

On considère ici un alphabet totalement ordonné de $k \geq 2$ symboles, noté Σ.

1 Mots de Lyndon

1.1 Définitions

Définition 1 (Mot) *Un mot est une suite finie de longueur $n \geq 0$ de symboles $m = m_0 \cdots m_{n-1}$ où, pour tout $i \in \{0, \ldots, n-1\}$, on a $m_i \in \Sigma$ et où n est la longueur de m, ce qu'on notera $n = |m|$.*
On note Σ^n l'ensemble des mots de longueur n construits sur Σ et Σ^* l'ensemble des mots de longueur quelconque construits sur Σ. On note enfin ε le mot vide.

Le type Caml choisi pour représenter les mots est le type *chaîne de caractères* (`string`). Les éléments de Σ sont représentés par le type *caractère* (`char`).

Dans toute la suite, les seules fonctions Caml sur les chaînes de caractères qui peuvent être utilisées sont :

- `s.[i]` : valeur du $i^{\text{ième}}$ caractère de s ;
- l'opérateur de concaténation $\wedge$;
- la fonction `String.length : string → int` : `String.length s` retourne la longueur de `s` ;
- la fonction `String.sub : string→ int→ int→ string` : `String.sub s start l` retourne une chaîne de longueur `l` contenant la sous-chaîne de `s` qui commence en `start`.

Définition 2 (Préfixe, suffixe) *Soient $m = m_0 \cdots m_{|m|-1}$ et $p = p_0 \cdots p_{|p|-1}$ deux mots de Σ^*. Alors :*

1. *$mp = m_0 \cdots m_{|m|-1} p_0 \cdots p_{|p|-1} \in \Sigma^{|m|+|p|}$ est le* **concaténé** *de m et p.*
2. *m est un* **préfixe** *strict de p si $|m| < |p|$ et $m_i = p_i$ pour $0 \leq i \leq |m| - 1$.*
3. *m est un* **suffixe** *strict de p si $|m| < |p|$ et $m_i = p_{|p|-|m|+i}$ pour $0 \leq i \leq |m| - 1$.*

On définit alors la relation d'ordre strict $\prec$ par :
$m \prec p \Leftrightarrow \Big(m \text{ est un préfixe strict de } p\Big)$ *ou* $\Big(\exists k \in [|0, |m| - 1|],\ \forall i \in [|0, k-1|],\ m_i = p_i \text{ et } m_k < p_k\Big)$,
où $<$ est la relation d'ordre strict naturelle sur Σ.

On donne le type

```
type comparaison = Inferieur | Egal | Superieur ;;
```

1. Écrire une fonction recursive `ordre : string → string → comparaison` telle que `ordre m p` est l'ordre relatif des mots m et p.

Définition 3 (Ordre lexicographique) *La relation d'ordre associée, définie par : $m \preccurlyeq p$ si et seulement si $m = p$ ou $m \prec p$ est appelée ordre lexicographique sur Σ^*.*

Définition 4 (Conjugué) *Soit $m \in \Sigma^n$.*
Un **conjugué** *de m est un mot de la forme $m_i \cdots m_{n-1} m_0 \cdots m_{i-1}$, pour $i \in [|0, n-1|]$.*
Le conjugué de m obtenu pour $i = 0$ est le mot m lui-même.

2. Écrire une fonction Caml `conjugue : string → int → string` telle que `conjugue m i` retourne le conjugué de `m` débutant par le $i^{\text{ième}}$ caractère de `m`, pour $0 \leq i \leq |m| - 1$.

La notion de conjugaison induit une relation $\mathcal{C}$ définie sur Σ^* par $m \,\mathcal{C}\, p$ si et seulement si p est un conjugué de m.

3. Montrer que $\mathcal{C}$ est une relation d'équivalence.

Définition 5 (Collier) *Un collier est le plus petit mot dans l'ordre lexicographique d'une classe d'équivalence pour la relation $\mathcal{C}$.*
Un collier d'ordre n est dit périodique s'il peut s'écrire m^l, où $m \in \Sigma^r$, $r \geq 1$ et $l > 1$.
Il est dit apériodique sinon, autrement dit si deux conjugaisons non triviales des membres de sa classe d'équivalence ne sont jamais égales.

Définition 6 (Mot de Lyndon) *Un mot $m \in \Sigma^*$ est un mot de Lyndon si c'est un collier apériodique.*

4. On suppose $\Sigma = \{0, 1\}$, avec $0 < 1$. Pour les mots suivants, indiquer si ce sont ou non des mots de Lyndon. Dans le cas négatif comme dans le cas positif, justifier votre réponse :
   i. 0010011 ;
   ii. 010011 ;
   iii. 001001.

5. Écrire une fonction Caml `Lyndon : string → bool` telle que `Lyndon m` renvoie `true` si `m` est un mot de Lyndon, et `false` sinon. Cette fonction fera appel à une fonction récursive.

1.2 Génération de mots de Lyndon

Soit $m \in \Sigma^*$ un mot de Lyndon. Pour générer à partir de m un mot de Lyndon q sur Σ de longueur au plus n avec $n \geq |m|$, on utilise **l'algorithme 1**, dont on admet qu'il calcule le plus petit mot de Lyndon qui soit $> m$.

Algorithme 1 : Algorithme de génération d'un mot de Lyndon

Données : $m \in \Sigma^*$ un mot de Lyndon, $n \geq |m|$.

Résultat : q le mot de Lyndon généré à partir de m.

*** Étape 1 ***

Concaténer le mot m à lui-même jusqu'à obtenir un mot q de longueur n. La dernière occurrence de m devra le cas échéant être tronquée pour arriver à un mot de longueur exactement n.

*** Étape 2 ***

Tant que le dernier symbole de q est le plus grand symbole de Σ faire :

↪ Ôter ce symbole de q.

*** Étape 3 ***

Remplacer le dernier symbole de q par le symbole qui suit dans Σ.

Retourner q**.**

6. Donner l'indice dans m de la $i^{\text{ième}}$ lettre de q en fonction de i et $|m|$.

7. Pour $\Sigma = \{0, 1\}$, on donne le mot de Lyndon $m = 00111$ et $n = 9$. Donner le mot de Lyndon généré par l'algorithme, en déroulant les différentes étapes produites par l'algorithme permettant d'aboutir au mot de Lyndon.

L'algorithme 1 peut être utilisé pour générer tous les mots de Lyndon de longueur au plus n. Pour ce faire, on part du plus petit mot de Lyndon $m^{(0)}$ qui est le mot de longueur 1 formé plus petit symbole de Σ. Puis on applique l'algorithme 1 au mot $m^{(0)}$ pour obtenir le mot de Lyndon suivant $m^{(1)}$. Puis on applique l'algorithme 1 au mot $m^{(1)}$ pour obtenir le mot de Lyndon suivant $m^{(2)}$. Le cas d'arrêt est celui où, en appliquant l'algorithme 1 à partir de $m^{(k)}$, on obtient le mot vide ε qui n'est pas un mot de Lyndon : le mot $m^{(k)}$ est alors le plus grand des mots de Lyndon.

8. Partant de $m = 0$ et toujours pour $\Sigma = \{0, 1\}$, construire par l'algorithme tous les mots de Lyndon de longueur au plus 4.

9. Donner la complexité de l'algorithme 1 dans le pire des cas, en nombre d'ajouts ou de suppressions de caractères.

1.3 Factorisation de mots de Lyndon

Définition 7 (Factorisation de Lyndon) *Soit $m \in \Sigma^*$. Une factorisation de m est une suite $m_1, \ldots, m_l$ de mots de Lyndon telle que $m = m_1 \ldots m_l$, avec $m_1 \succcurlyeq m_2 \succcurlyeq \cdots \succcurlyeq m_l$.*

On admet le résultat suivant :

Théorème 1 (Factorisation d'un mot) *Tout mot $m \in \Sigma^*$ admet une unique factorisation de Lyndon.*

L'algorithme 2 propose une méthode de factorisation d'un mot m (on ne demande pas de le justifier).

Le principe est d'itérer sur la chaîne des symboles de m pour trouver le plus grand mot de Lyndon possible. Lorsqu'un tel mot est trouvé, il est ajouté à la liste $\mathcal{L}$ des facteurs de m et la recherche est poursuivie sur la sous-chaîne restante.

10. En utilisant l'algorithme 2, écrire une fonction `factorisation : string` $\to$ `string list` qui réalise la factorisation d'un mot donné. Cette fonction fera appel à une ou des fonction(s) récursive(s).

Algorithme 2 : Algorithme de factorisation d'un mot de Lyndon

---

Données : $m \in \Sigma^*$ un mot.

Résultat : la liste $\mathcal{L}$ des mots de Lyndon décroissants de la factorisation de m.

$\mathcal{L} \leftarrow []$

$j \leftarrow 1$

$k \leftarrow 0$

Tant que $j \leq |m|$:

↪ **Si** $j = |m|$ ou $m_k > m_j$ alors :

↪ $p = m_0 \cdots m_{j-k-1}$;

Ajouter p à $\mathcal{L}$;

Supprimer p de m ;

$k \leftarrow 0$;

$j \leftarrow 1$.

↪ **Sinon, si** $m_k = m_j$ alors :

↪ $k \leftarrow k + 1$;

$j \leftarrow j + 1$.

↪ **Sinon** :

↪ $k \leftarrow 0$;

$j \leftarrow j + 1$.

Retourner $\mathcal{L}$.

2 Mots de de Bruijn

2.1 Définitions

Définition 8 (Mot de de Bruijn) *Un mot de de Bruijn d'ordre n sur Σ est un collier qui contient tous les mots de Σ^n une et une seule fois.*

Par exemple, pour $\Sigma = \{a, b, c\}$ et $n = 2$, $m = aabacbbcc$ est un mot de de Bruijn puisque c'est bien un collier et qu'il contient une unique fois chaque mot de longueur 2 sur Σ, le mot ca étant obtenu par circularité de m.

11. Donner la longueur d'un mot de de Bruijn en fonction de n et du nombre k de symboles de Σ.

2.2 Graphe de de Bruijn

Définition 9 (Graphe de de Bruijn) *Le graphe de de Bruijn d'ordre n sur Σ est le graphe orienté $B(k, n) = (V, E)$ où :*

- *$V = \Sigma^n$ est l'ensemble des sommets du graphe ;*
- *$E = \left\{(am, mb), (a, b, m) \in \Sigma \times \Sigma \times \Sigma^{n-1}\right\}$ est l'ensemble des arcs orientés du graphe.*

De plus, on value les arcs E par le dernier symbole du noeud terminal de chaque arc : ainsi $(am, mb) \in E$ est étiqueté par b. Attention, dans ce graphe (comme sur la figure 1), certains arcs ont pour sommet initial et terminal un même sommet de V. Ces arcs sont appelés des boucles.

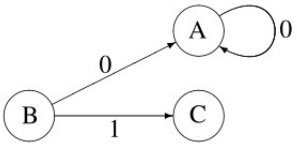

Figure 2.- Exemple de graphe avec boucle sur le sommet A.

12. Donner l'ensemble des mots de longueur 3 sur $\Sigma = \{0, 1\}$. En déduire de graphe de de Bruijn $B(2, 3)$ associé.

13. Montrer que le degré entrant et le degré sortant de chaque sommet est égal à k.

14. En déduire le nombre d'arcs orientés $|E|$ en fonction de k et n.

Définition 10 (Successeur, prédécesseur) *Soit $G = (V, E)$ un graphe orienté.*
Un sommet $v \in V$ est un successeur (respectivement prédécesseur) de $u \in V$ si $(u, v) \in E$ (resp. $(v, u) \in E$).

15. Soient $B(k, n) = (V, E)$ et $m \in V$. Montrer que tous les prédécesseurs de m ont le même ensemble de successeurs.

16. Soit $p = p_0 \cdots p_{n-1} \in V$. Donner l'ensemble des sommets m tels que $(p, m) \in E$.

On suppose disposer de $B(k, n)$ et on souhaite construire $B(k, n + 1)$ à partir de ce graphe.

17. Proposer une méthode pour construire les sommets de B(k,n + 1) à partir des arcs de B(k,n).
Donner le sommet créé dans B(2,4) à partir de l'arc (001,010) de B(2,3).

On dit que deux arcs $e_1, e_2 \in E$ sont adjacents dans $B(k, n)$ s'ils s'écrivent $e_1 = (m, p)$ et $e_2 = (p, q)$ pour $(m, p, q) \in V^3$.

18. Proposer de même une construction des arcs de$B(k, n + 1)$ en fonction des arcs adjacents de $B(k, n)$.
Donner l'arc de $B(2, 4)$ créé par les deux arcs $(001, 011)$ et $(011, 110)$, arcs adjacents de $B(2, 3)$.

2.3 Construction des mots de de Bruijn

On propose ici trois algorithmes de construction de mots de de Bruijn.

2.3.1 Construction à l'aide de B(k,n)

Les mots de de Bruijn d'ordre n sur Σ peuvent être construits en parcourant $B(k, n)$.

Définition 11 (Circuit eulérien) *Soit G un graphe orienté. Un circuit eulérien est un chemin dont l'origine et l'extrémité coïncident et passant une fois et une seule par chaque arête de G.*

19. Construire $B(2, 2)$ et trouver dans ce graphe un circuit eulérien. Vérifier que la concaténation des étiquettes lues au fil de ce circuit donne un représentant d'un mot de de Bruijn et donner son ordre sur $\{0, 1\}$.

On admet les résultats suivants :

i. Un graphe $G = (V, E)$ possède un circuit eulérien si et seulement si il est connexe et que, pour tout $v \in V$, le degré entrant de v et le degré sortant de v sont égaux.

ii. un circuit eulérien dans le graphe $B(k, n)$ correspond à un mot de de Bruijn.

20. En déduire qu'il existe au moins un mot de de Bruijn pour tout alphabet Σ et tout entier n. Ainsi, la concaténation des étiquettes lues au fil d'un circuit eulérien de $B(k, n)$ donne un conjugué d'un mot de de Bruijn d'ordre $n + 1$ sur k symboles.

2.3.2 Construction à l'aide de l'algorithme Prefer One

Pour construire les mots de de Bruijn d'ordre n sur $\Sigma = \{0, 1\}$, on peut également utiliser l'algorithme 3, dit algorithme Prefer One.

Algorithme 3 : Algorithme Prefer One

Données : n, Σ.

Résultat : m, mot de de Bruijn de longueur n sur Σ.

$m \leftarrow$ suite de n zeros

STOP $\leftarrow$ false

Tant que non(STOP) faire :

↪ *** Étape 1 ***

Ajouter un 1 à la fin de m.

Si : les n derniers symboles de m n'ont pas été rencontrés auparavant alors :

↪ Répeter l'étape 1.

Sinon :

↪ Retirer le 1 ajouté à la fin de m.

Passer à l'étape 2

*** Étape 2 ***

Ajouter un 0 à la fin de m.

Si : les n derniers symboles de m n'ont pas été rencontrés auparavant alors

↪ Aller à l'étape 1.

Sinon :

↪ STOP $\leftarrow$ true

Retourner m.

Dans cet algorithme, l'expression "les n derniers symboles de m n'ont pas été rencontrés auparavant" signifie que le mot composé des n derniers symboles de m n'est pas un sous-mot de m, situé entre le premier et le $(|m|-1)^{\text{ième}}$ symbole de m.

21. Appliquer l'algorithme au cas $n = 3$ et $\Sigma = \{0, 1\}$. Écrire les valeurs successives de m au cours de l'exécution.

2.3.3 Construction à l'aide de la relation aux mots de Lyndon

La troisième construction utilise les notions de collier et de mots de Lyndon.

22. Donner, pour $k = 2$ et $n = 4$, les classes d'équivalence de tous les mots binaires de longueur 4 pour la relation $\mathcal{C}$. En déduire les colliers correspondants.

23. En déduire les mots de Lyndon de longueur 4 pour $\Sigma = \{0, 1\}$.

Plus généralement, les mots de de Bruijn et de Lyndon sont étroitement liés. On peut montrer que si l'on concatène, dans l'ordre lexicographique, les mots de Lyndon sur k symboles dont la longueur divise un entier n, alors on obtient le mot de de Bruijn le plus petit, pour l'ordre lexicographique, de tous les mots de de Bruijn de longueur n sur k symboles.

24. En déduire, à l'aide de la question précédente, le plus petit mot de de Bruijn pour $n = 4$ et $\Sigma = \{0, 1\}$.

3 Application

La soirée a été longue, vous rentrez chez vous mais, au pied de votre immeuble, vous vous trouvez confronté à un sérieux problème : vous avez complètement oublié le code d'entrée à n chiffres de la porte.

On suppose que le digicode de l'appartement est composé de $1 \leq k \leq 10$ chiffres $0, 1, \ldots, k-1$, qui constituent l'alphabet Σ.

Ce digicode fonctionne de la façon suivante : vous tapez successivement sur les chiffres afin de composer un mot. À chaque nouveau symbole entré à partir du n-ième, le digicode teste le mot constitué par les n derniers chiffres pour voir s'il correspond au code secret. Ainsi, par exemple pour $n = 4$, si vous tapez la séquence 021201, le digicode teste successivement 0212, 2120 et 1201.

Étant pressé de regagner votre lit, vous cherchez à taper un minimum de touches pour ouvrir la porte. On note $\tilde{n}$ la longueur de la plus petite séquence de frappe de touches qui permette d'être certain de rentrer dans l'immeuble.

25. Donner un encadrement de $\tilde{n}$:
    - pour la borne supérieure, on considèrera que l'on met bout à bout tous les mots possibles de n chiffres construits sur Σ ;
    - pour la borne inférieure, on cherchera un mot sans redondance, c'est-à-dire qui contient une et une seule fois chaque mot de n chiffres.

26. Expliquer en une phrase en quoi les mots de de Bruijn peuvent vous aider. En vous inspirant de l'algorithme 3, donner une séquence la plus courte de chiffres à taper pour ouvrir à coup sûr la porte de votre immeuble, lorsque $n = 2$ et $k = 4$.

27. Comparer alors le nombre maximum de frappes de touches du digicode à effectuer en utilisant les mots de de Bruijn, avec celui calculé par la méthode brute. Calculer ces nombres pour :
    - $k = 4$ et $n = 2$;
    - $k = 10$ et $n = 4$.